A Construction of a Family of RSA Functions with a Common Domain (Evolutionary Advancement in Fundamental Theories of Computer Science)

AUTHOR(S):

Hayashi, Ryotaro; Tanaka, Keisuke

数理解析研究所講究録 1375 巻 2004 年 164-170

同じ値域をもつRSA関数族の構成
A Construction of a Family of RSA Functions with a Common Domain

林 良太郎　　田中 圭介
Ryotaro Hayashi　　Keisuke Tanaka

東京工業大学 数理・計算科学専攻
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology

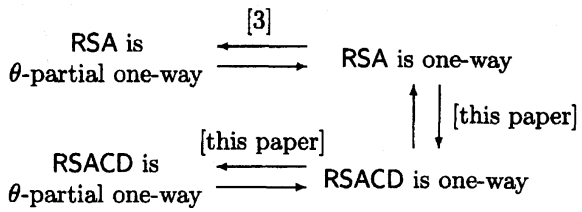
Abstract— For a standard RSA family of trap-door permutations, even if all of the functions in a family use RSA moduli of the same size (the same number of bits), it will have domains with different sizes. In this paper, we construct an RSA family of trap-door permutations with a common domain. We also construct a family of Paillier's trap door permutations with a common domain.

Keywords: trap-door permutations, RSA, Paillier's permutations

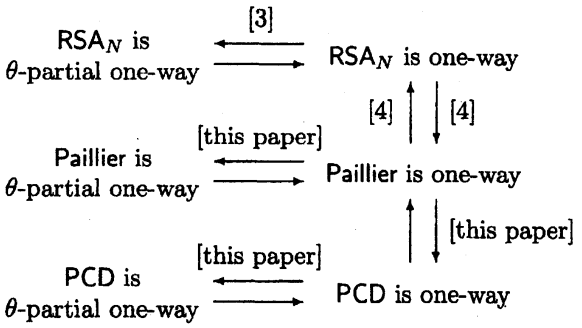
1 Introduction

Bellare, Boldyreva, Desai, and Pointcheval [1] recently proposed a new security requirement of the encryption schemes called "key-privacy." It asks that the encryption provide (in addition to privacy of the data being encrypted) privacy of the key under which the encryption was performed. The standard RSA encryption does not provide key-privacy. Since even if two public keys N_0 and N_1 ($N_0 < N_1$) are the same bits, $N_1 - N_0$ may be large. In [1], they provided the key-privacy encryption scheme, RSA-RAEP, which is a variant of RSA-OAEP (Bellare and Rogaway [2], Fujisaki, Okamoto, Pointcheval, and Stern [3]), and solved this problem by repeating the evaluation of the RSA-OAEP permutation $f(x, r)$ with plaintext x and random r, each time using different r until the value is in the safe range.

We are concerned with an underlying primitive element, that is, families of trap-door permutations with a common domain. For a standard RSA family of trap-door permutations denoted by RSA, even if all of the functions in a family use RSA moduli of the same size (the same number of bits), it will have domains with different sizes. We construct an RSA family of trap-door permutations with a common domain denoted by RSACD, and prove that the θ-partial one-wayness of RSACD is equivalent to the one-wayness of RSACD for $\theta > 0.5$, and that the one-wayness of RSACD is equivalent to the one-wayness of RSA. Thus, the following relations are satisfied for $\theta > 0.5$.



In [4], Paillier provided a trap-door one-way bijective function, and proved that the function is one-way if and only if RSA$[N, N]$ is hard, where RSA$[N, N]$ is the problem of extracting N-th roots modulo N. We slightly modified his function and construct the family of Paillier's trap-door permutations. We also construct the family of Paillier's trap-door permutations with a common domain denoted by PCD, and prove the following relations for $\theta > 0.5$.



where Paillier denotes a family of Paillier's trap-door permutations and RSA$_N$ denotes an RSA family of trap-door permutations with the fixed exponent N.

2 An RSA Family of Trap-door Permutations with a Common Domain

2.1 Preliminaries

In this section, we briefly review the definitions of families of functions, and the standard RSA family of trap-door permutations denoted by RSA.

Definition 1 (families of functions [1]). *A family of functions $F = (K, S, E)$ is specified by three algorithms.*

- *The randomized key-generation algorithm K takes as input a security parameter $k \in \mathbb{N}$ and returns a pair (pk, sk) where pk is a public key and sk is an associated secret key. (In cases where the family is not trap-door, the secret key is simply the empty string.)*
- *The randomized sampling algorithm S takes input pk and returns a random point in a set that we call the domain of pk and denote by $\mathrm{Dom}_F(pk)$.*
- *The deterministic evaluation algorithm E takes input pk and a point $x \in \mathrm{Dom}_F(pk)$ and returns an output we denote by $E_{pk}(x)$. We let $\mathrm{Rng}_F(pk) = \{E_{pk}(x) \mid x \in \mathrm{Dom}_F(pk)\}$ denote the range of the function $E_{pk}(\cdot)$.*

Definition 2 (families of trap-door permutations [1]). *We say that F is a family of trap-door functions if there exists a deterministic inversion algorithm I that takes input sk and a point $y \in \mathrm{Rng}_F(pk)$ and returns a point $x \in \mathrm{Dom}_F(pk)$ such that $E_{pk}(x) = y$. We say that F is a family of trap-door permutations if F is a family of trap-door functions, $\mathrm{Dom}_F(pk) = \mathrm{Rng}_F(pk)$, and E_{pk} is a permutation on this set.*

We describe the definition of θ-partial one-way.

Definition 3 (θ-partial one-way [1]). *Let $F = (K, S, E)$ be a family of functions. Let $b \in \{0,1\}$ and $k \in \mathbb{N}$ be a security parameter. Let $0 < \theta \leq 1$ be a constant. Let A be an adversary. Now, we consider the following experiments:*

Experiment $\mathbf{Exp}^{\theta\text{-pow-fnc}}_{F,A}(k)$
$(pk, sk) \xleftarrow{R} K(k)$
$x_1||x_2 \xleftarrow{R} \mathrm{Dom}_F(pk)$ **where** $|x_1| = \lceil \theta \cdot |(x_1||x_2)| \rceil$
$y \leftarrow E_{pk}(x_1||x_2)$
$x'_1 \leftarrow A(pk, y)$ **where** $|x'_1| = |x_1|$
for any x'_2 **if** $E_{pk}(x'_1||x'_2) = y$ **then return** 1
else return 0

We define the advantages of the adversary via

$$\mathbf{Adv}^{\theta\text{-pow-fnc}}_{F,A}(k) = \Pr[\mathbf{Exp}^{\theta\text{-pow-fnc}}_{F,A}(k) = 1]$$

where the probability is taken over $(pk, sk) \xleftarrow{R} K(k)$, $x_1||x_2 \xleftarrow{R} \mathrm{Dom}_F(pk)$, and the coin tosses of A. We say that the family F is θ-partial one-way if the function $\mathbf{Adv}^{\theta\text{-pow-fnc}}_{F,A}(\cdot)$ is negligible for any adversary A whose time complexity is polynomial in k. In particular, we say that the family F is one-way when F is 1-partial one-way.

We describe the standard RSA family of trap-door permutations denoted by RSA.

Definition 4 (the standard RSA family of trap-door permutations). *The specifications of the standard RSA family of trap-door permutations $\mathsf{RSA} = (K, S, E)$ are as follows. The key generation algorithm takes as input a security parameter k and picks random, distinct primes p, q in the range $2^{\lceil k/2 \rceil - 1} < p, q < 2^{\lceil k/2 \rceil}$ and $2^{k-1} < N < 2^k$. It sets $N = pq$. It picks $e, d \in \mathbb{Z}^*_{\phi(N)}$ such that $ed = 1 \pmod{\phi(N)}$ where $\phi(N) = (p-1)(q-1)$. The public key is N, e, k and the secret key is N, d, k. The sets $\mathrm{Dom}_{\mathsf{RSA}}(N, e, k)$ and $\mathrm{Rng}_{\mathsf{RSA}}(N, e, k)$ are both equal to $\mathbb{Z}^*_N$. The evaluation algorithm $E_{N,e,k}(x) = x^e \bmod N$ and the inversion algorithm $I_{N,d,k}(y) = y^d \bmod N$. The sampling algorithm returns a random point in $\mathbb{Z}^*_N$. The sampling algorithm returns a random point in $\mathbb{Z}^*_N$.*

Fujisaki, Okamoto, Pointcheval, and Stern [3] showed that the θ-partial one-wayness of RSA is equivalent to the one-wayness of RSA for $\theta > 0.5$.

2.2 The Construction of RSACD

In this section, we propose the RSA family of trap-door permutations with a common domain denoted by RSACD.

Definition 5 (the RSA family of trap-door permutations with a common domain). *The specifications of the RSA family of trap-door permutations with a common domain $\mathsf{RSACD} = (K, S, E)$ are as follows. The key generation algorithm is the same as that for RSA. The sets $\mathrm{Dom}_{\mathsf{RSACD}}(N, e, k)$ and $\mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$ are both $\{x \mid x \in [0, 2^k) \wedge x \bmod N \in \mathbb{Z}^*_N\}$. The sampling algorithm returns a random point in $\mathrm{Dom}_{\mathsf{RSACD}}(N, e, k)$. The evaluation algorithm $E_{N,e,k}(x) = f_{N,e,k}(x)$ and the inversion algorithm $I_{N,d,k}(y) = g_{N,d,k}(y)$ are as follows (See Figure 1.).*

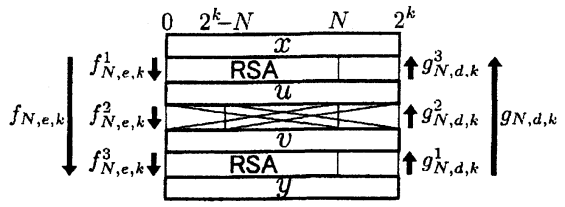


Figure 1: Function $f_{N,e,k}$ and $g_{N,d,k}$

```
Function f_{N,e,k}(x)
  u ← f^1_{N,e,k}(x); v ← f^2_{N,e,k}(u); y ← f^3_{N,e,k}(v)
  return y

Function f^1_{N,e,k}(x)
  if (x < N) u ← x^e mod N else u ← x
  return u

Function f^2_{N,e,k}(u)
  if (u < 2^k − N) v ← u + N
  elseif (2^k − N ≤ u < N) v ← u
  else v ← u − N
  return v

Function f^3_{N,e,k}(v)
  if (v < N) y ← v^e mod N else y ← v
  return y

Function g_{N,d,k}(y)
  v ← g^1_{N,d,k}(y); u ← g^2_{N,d,k}(v) x ← g^3_{N,d,k}(u)
  return x

Function g^1_{N,d,k}(y)
  if (y < N) v ← y^d mod N else v ← y
  return v

Function g^2_{N,d,k}(v)
  if (v < 2^k − N) u ← v + N
  elseif (2^k − N ≤ v < N) u ← v
  else u ← v − N
  return u

Function g^3_{N,d,k}(u)
  if (u < N) x ← u^d mod N else x ← u
  return x
```

The choice of N from $(2^{k-1}, 2^k)$ ensures that all elements in $\mathrm{Dom}_{\mathsf{RSACD}}(N, e, k)$ are permuted by the RSA function at least once.

2.3 Properties of RSACD

In this section, we prove that the θ-partial one-wayness of RSACD is equivalent to the one-wayness of RSACD for $\theta > 0.5$, and that the one-wayness of RSACD is equivalent to the one-wayness of RSA.

Theorem 1. *If* RSACD *is one-way then* RSACD *is θ-partial one-way for $\theta > 0.5$.*

To prove this theorem, we use the following lemma proved in [3].

Lemma 1 ([3]). *Consider an equation $\alpha t + u = c \pmod N$ which has solutions t and u smaller than 2^{k_0}. For all values of α, except a fraction $2^{2k_0+6}/N$ of them, (t, u) is unique and can be computed in time $O((\log N)^3)$. (We say "α is a good value" when we can solve the above equation.)*

Proof of Theorem 1. Let A be an algorithm that outputs the $k - k_0$ most significant bits of the pre-image of its input $y \in \mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$ for $2^{k-1} < N < 2^k$ with $k > 2k_0$ (i.e. A is a $((k-k_0)/k)$-partial inverting algorithm for RSACD with $k > 2k_0$), with success probability $\epsilon = \mathbf{Adv}^{\theta\text{-pow-fnc}}_{\mathsf{RSACD},A}(k)$ where $\theta = (k - k_0)/k > 0.5$, within time bound t. There exists an algorithm B that outputs a pre-image of y (i.e. B is an inverting algorithm for RSACD) with success probability $\epsilon' = \mathbf{Adv}^{1\text{-pow-fnc}}_{\mathsf{RSACD},B}(k)$, within time bound t' where

$$\epsilon' \geq \frac{\epsilon^2}{16} \cdot (1 - 2^{2k_0-k+7}), \quad t' \leq 2t + O(k^3).$$

We construct the algorithm B to compute a pre-image of $y \in \mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$, then we analyze this algorithm and evaluate the success probability and the running time of B.

```
Algorithm B((N, e, k), y)

  %%% [step 1] set α, pow, y' %%%
  α ←R Z_N;  pow ←R {1, 2};  c ←R {0, 1}
  y'_temp ← y · α^{e^pow} mod N
  if (c = 0) y' ← y'_temp
  elseif (0 ≤ y'_temp < 2^k − N) y' ← y'_temp + N
  else return fail

  %%% [step 2] run A %%%
  z ← A(y); z' ← A(y')

  %%% [step 3] compute g_{N,d,k}(y) %%%
  find (r, s) s.t.  αr − s = (z' − zα) · 2^{k_0}  (mod N)
  x ← z · 2^{k_0} + r

  return x
```

Analysis

For $y \in \mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$ and $x = g_{N,d,k}(y)$, (x, y) satisfies one of the following equations.

$$(1) \quad y = x^e \pmod N$$
$$(2) \quad y = x^{e^2} \pmod N$$

We say $type(y) = 1$ (respectively $type(y) = 2$) if (x, y) satisfies equation 1 (resp. equation 2).

After step 1, if B does not output `fail`, then y' is uniformly distributed over $\mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$, and for y' and $x' = g_{N,d,k}(y')$, (x', y') satisfies one of the following equations.

$$(1') \quad y' = (x')^e \pmod N$$
$$(2') \quad y' = (x')^{e^2} \pmod N$$

We say $type(y') = 1$ (respectively $type(y') = 2$) if (x', y') satisfies equation 1' (resp. equation 2').

After step 2, if A outputs correctly, namely, z is the $k - k_0$ most significant bits of x and x' is the $k-k_0$ most significant bits of x', then $x = z \cdot 2^{k_0} + r$ and $x' = z' \cdot 2^{k_0} + s$ for some (r, s) where $0 \le r, s < 2^{k_0}$. Furthermore, if $type(y) = type(y') = \mathrm{pow}$, then $y = x^{e^{\mathrm{pow}}} \pmod N$ and $y' = (x')^{e^{\mathrm{pow}}} \pmod N$. Since $y' = y \cdot \alpha^{e^{\mathrm{pow}}} \pmod N$ and $\gcd(e^{\mathrm{pow}}, N) = 1$, we have $x' = \alpha x \pmod N$. Thus,

$$z' \cdot 2^{k_0} + s = \alpha \cdot (z \cdot 2^{k_0} + r) \pmod N$$
$$\alpha r - s = (z' - z\alpha) \cdot 2^{k_0} \pmod N$$

where $0 \le r, s < 2^{k_0}$. If α is a good value, algorithm B can solve this equation in step 3 (Lemma 1), and outputs $x = z \cdot 2^{k_0} + r$.

Now, we analyze the success probability. We define the following events:

- Fail : B outputs `fail` in step 1,
- GV : α is a good value,
- Type1 : $type(y) = type(y') = 1$,
- Type2 : $type(y) = type(y') = 2$,
- SucA : $A(y)$ and $A(y')$ are correct.

We have $\epsilon = \Pr[A(y) \text{ is correct} \wedge type(y) = 1] + \Pr[A(y) \text{ is correct} \wedge type(y) = 2]$ where y is uniformly distributed over $\mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$. Thus, $\Pr[A(y) \text{ is correct} \wedge type(y) = 1] > \epsilon/2$ or $\Pr[A(y) \text{ is correct} \wedge type(y) = 2] > \epsilon/2$.
If B does not output `fail` in step 1, then y' is uniformly distributed over $\mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$. Therefore, $\Pr[\mathsf{SucA} \wedge \mathsf{Type1} | \neg\mathsf{Fail}] > (\epsilon/2)^2 = \epsilon^2/4$ or $\Pr[\mathsf{SucA} \wedge \mathsf{Type2} | \neg\mathsf{Fail}] > (\epsilon/2)^2 = \epsilon^2/4$.
If $A(y)$ and $A(y')$ are correct, $type(y) = type(y') = \mathrm{pow}$, and α is a good value, then B outputs correctly. Since $\Pr[\neg\mathsf{Fail}] > \Pr[c = 1] = 1/2$, $\Pr[\mathrm{pow} = 1] = \Pr[\mathrm{pow} = 2] = 1/2$, and $\Pr[\mathsf{GV}] > 1 - 2^{2k_0-6}/N > 1 - 2^{2k_0-k+7}$, we have

$$\begin{aligned}
\epsilon' &\ge \Pr[\mathsf{SucA} \wedge type(y) = type(y') = \mathrm{pow} \wedge \mathsf{GV}] \\
&\ge \Pr[\mathsf{GV}] \times \Pr[\neg\mathsf{Fail}] \times \\
&\quad \Pr[\mathsf{SucA} \wedge type(y) = type(y') = \mathrm{pow} | \neg\mathsf{Fail}] \\
&\ge \frac{1}{2} \cdot (1 - 2^{2k_0-k+7}) \times \\
&\quad (\Pr[\mathsf{SucA} \wedge \mathsf{Type1} \wedge \mathrm{pow} = 1 | \neg\mathsf{Fail}] \\
&\quad + \Pr[\mathsf{SucA} \wedge \mathsf{Type2} \wedge \mathrm{pow} = 2 | \neg\mathsf{Fail}]) \\
&= \frac{1}{2} \cdot (1 - 2^{2k_0-k+7}) \times \\
&\quad (\Pr[\mathrm{pow} = 1] \times \Pr[\mathsf{SucA} \wedge \mathsf{Type1} | \neg\mathsf{Fail}] \\
&\quad + \Pr[\mathrm{pow} = 2] \times \Pr[\mathsf{SucA} \wedge \mathsf{Type2} | \neg\mathsf{Fail}]) \\
&= \frac{\epsilon^2}{16} \cdot (1 - 2^{2k_0-k+7}).
\end{aligned}$$

We estimate the running time of B. B runs A twice. B can solve $\alpha r - s = (z' - z\alpha) \cdot 2^{k_0} \pmod N$ in time $O(k^3)$. Therefore, $t' \le 2t + O(k^3)$. □

Theorem 2. *If* RSA *is one-way then* RSACD *is one-way.*

Proof. We prove that if there exists a poly-time inverting algorithm A for RSACD with non-negligible probability $\epsilon = \mathbf{Adv}^{1-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{RSACD},A}(k)$, then there exists a poly-time inverting algorithm D for RSA with non-negligible probability $\epsilon' = \mathbf{Adv}^{1-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{RSA},D}(k)$. We show the algorithm D to compute a pre-image of $Y \in \mathrm{Rng}_{\mathsf{RSA}}(N, e, k)$.

```
Algorithm D((N, e, k), Y)
  c ←R {0, 1}
  if (c = 0)
    y ← Y; x ← A((N, e, k), y); u ← f^1_{N,e,k}(x)
    v ← f^2_{N,e,k}(u); X ← v
  else
    u ← Y; v ← f^2_{N,e,k}(u); y ← f^3_{N,e,k}(v)
    x ← A((N, e, k), y); X ← x
  return X
```

Now, we analyze the advantage of D. If A outputs correctly then D outputs correctly (See Figure 1). Therefore,

$$\begin{aligned}
\epsilon' &> \Pr[\neg\mathsf{Fail}] \cdot (\Pr[c = 0 \ \wedge \ A((N, e, k), Y) \text{ is correct}] \\
&\quad + \Pr[c = 1 \ \wedge \ A((N, e, k), Z) \text{ is correct}]) \\
&\ge \frac{1}{2} \cdot (\Pr[A((N, e, k), Y) \text{ is correct}] \\
&\quad + \Pr[A((N, e, k), Z) \text{ is correct} \ \wedge \ N \le Z < 2^k]).
\end{aligned}$$

where $Z = f^3_{N,e,k}(f^2_{N,e,k}(Y))$. We have

$$\begin{aligned}
&\Pr[A((N, e, k), Y) \text{ is correct}] \\
&= \Pr[A((N, e, k), y) \text{ is correct} \,|\, 0 \le y < N] \\
&> \Pr[A((N, e, k), y) \text{ is correct} \ \wedge \ 0 \le y < N].
\end{aligned}$$

Furthermore, we have $\Pr[N \le Z < 2^k] > \Pr[N \le y < 2^k]$ where Y is uniformly distributed over $\mathbb{Z}^*_N$ and y is uniformly distributed over $\mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)$, since $\Pr[N \le Z < 2^k] = \Pr[0 \le Y < 2^k - N]$ and $|\mathbb{Z}^*_N| < |\mathrm{Rng}_{\mathsf{RSACD}}(N, e, k)|$.

Since $\Pr[A((N,e,k),Z) \text{ is correct} \mid N \leq Z < 2^k] = \Pr[A((N,e,k),y) \text{ is correct} \mid N \leq y < 2^k]$, we have

$$\Pr[A((N,e,k),Z) \text{ is correct} \wedge N \leq Z < 2^k] > \Pr[A((N,e,k),y) \text{ is correct} \wedge N \leq y < 2^k].$$

Therefore,

$$\begin{aligned}\epsilon' &> \frac{1}{2} \cdot (\Pr[A((N,e,k),y) \text{ is correct} \wedge 0 \leq y < N] \\ &\quad + \Pr[A((N,e,k),y) \text{ is correct} \wedge N \leq y < 2^k]) \\ &= \frac{1}{2} \cdot \Pr[A((N,e,k),y) \text{ is correct}] = \frac{1}{2} \cdot \epsilon\end{aligned}$$

which is non-negligible in k. □

It is clear that if RSACD is one-way then RSA is one-way. Thus, the one-wayness of RSACD is equivalent to the one-wayness of RSA.

3 A Family of Paillier's Trap-door Permutations with a Common Domain

3.1 Paillier's Trap-door Permutations

In [4], Paillier provided the trap-door one-way bijective function. He proved that his function is one-way if and only if RSA$[N,N]$ is hard, where RSA$[N,N]$ is the problem of extracting N-th roots modulo N. We slightly modify his function, and then, we consdier the family of Paillier's trap-door permutations denoted by Paillier.

Definition 6 (the family of Paillier's trap-door permutations). *The specifications of the family of Paillier's trap-door permutations* Paillier $= (K,S,D)$ *are as follows. The key generation algorithm K takes as input a security parameter k and picks random, distinct primes p,q such that $2^{\lceil k/2 \rceil - 1} < p,q < 2^{\lceil k/2 \rceil}$, $2^{k-1} < pq < 2^k$ and $2^{2k-1} < (pq)^2 < 2^{2k}$. It sets $N = pq$ and $\lambda = \lambda(N) = \mathrm{lcm}(p-1, q-1)$. The public key is N,k and the secret key is N,k,λ.* $\mathrm{Dom}_{\mathsf{Paillier}}(N,k)$ *and* $\mathrm{Rng}_{\mathsf{Paillier}}(N,k,\lambda)$ *are both equal to $\{x_1 + x_2 \cdot N | x_1 \in \mathbb{Z}_N, x_2 \in \mathbb{Z}_N^*\}$. The sampling algorithm returns a random point in* $\mathrm{Dom}_{\mathsf{Paillier}}(N,k)$. *The evaluation algorithm $E_{N,k}(x) = F_P(x)$, and the inversion algorithm $I_{N,k,\lambda}(y) = G_P(y)$ are as follows.*

```
Function F_P(x)
  x_1 ← x mod N;  x_2 ← x div N
  Y ← (1 + N x_1) x_2^N mod N^2
  y_1 ← Y div N;  y_2 ← Y mod N
  y ← y_1 + y_2 · N
  return y
```

```
Function G_P(y)
  y_1 ← y mod N;  y_2 ← y div N
  Y ← y_1 · N + y_2
  x_1 ← (L(Y^λ mod N^2) / λ) mod N
  y' ← y · (1 − N x_1) mod N^2
  x_2 ← (y')^{N^{-1} mod λ} mod N^2
  x ← x_1 + x_2 · N
  return x
```

We describe the RSA family of trap-door permutations with the fixed exponent N.

Definition 7 (the RSA family of trap-door permutations with the fixed exponent N). *The specifications of the RSA family of trap-door permutations with the fixed exponent N* $\mathsf{RSA}_N = (K,S,E)$ *are as follows. The key generation algorithm K is the same as that for* Paillier. $\mathrm{Dom}_{\mathsf{RSA}_N}(N,k)$ *and* $\mathrm{Rng}_{\mathsf{RSA}_N}(N,k,\lambda)$ *are both equal to $\mathbb{Z}_N^*$. The evaluation algorithm $E_{N,k}(x) = x^N \bmod N$ and the inversion algorithm $I_{N,k,\lambda}(y) = y^{N^{-1} \bmod \lambda} \bmod N$. The sampling algorithm returns a random point in $\mathbb{Z}_N^*$.*

Then, we can easily see the following lemma.

Lemma 2. Paillier *is one-way if and only if* RSA_N *is one-way.*

We prove the following theorem.

Theorem 3. *θ-partial one-wayness of* Paillier *is equivalent to the one-wayness of* Paillier *for $\theta > 0.5$.*

Proof. Let A be an algorithm that outputs the $2k - k_0$ most significant bits of the pre-image of its input $y \in \mathrm{Rng}_{\mathsf{Paillier}}(N,k)$ with $k > k_0$ (i.e. A is a $((2k-k_0)/2k)$-partial inverting algorithm for Paillier with $k > k_0$), with success probability $\epsilon = \mathbf{Adv}^{\theta-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{Paillier},A}(k)$ where $\theta = (2k-k_0)/k > 0.5$, within time bound t. We prove that there exists an algorithm B that outputs a pre-image of y with success probability $\epsilon' = \mathbf{Adv}^{1-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{Paillier},B}(k) \geq \epsilon/2$, within time bound $t' \leq t + O(k^3)$. We construct the algorithm B as follows.

```
Algorithm B((N,k),y)
  X ← A((N,k),y)
  c ←R {0,1}
  x_2 ← ((2^{k_0} · X) div N) + c

  y_1 ← y mod N;  y_2 ← y div N
  Y ← y_1 · N + y_2
  find x_1 s.t. 1 + N x_1 = Y / (x_2)^N mod N^2
  x ← x_1 + x_2 · N
  return x
```

Assume that A outputs correctly, that is, X is the most $2k - k_0$ significant bits of x. We know $x = 2^{k_0} \cdot X + R$ for some $0 < R < 2^{k_0}$. Thus, $x_2 = x \operatorname{div} N = ((2^{k_0} \cdot X) \operatorname{div} N) + ((((2^{k_0} \cdot X) \bmod N) + R) \operatorname{div} N)$. Since $R < 2^{k_0} \leq 2^{k-1} < N$ (Note that $k_0 \leq k - 1$, since $k, k_0 \in \mathbb{N}$ and $k_0 < k$.), we have $((2^{k_0} \cdot X) \bmod N) + R < 2N$. Hence, $((2^{k_0} \cdot X) \bmod N + R) \operatorname{div} N$ is equal to 0 or 1, and we have $x_2 = (2^{k_0} \cdot X) \operatorname{div} N$ or $(2^{k_0} \cdot X) \operatorname{div} N + 1$.

It is easy to see that if x_2 is correct then $x = x_1 + x_2 \cdot N$ is the pre-image of y. Therefore, $\epsilon' = \mathbf{Adv}^{1-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{Paillier},B}(k) \geq \epsilon/2$. It is easy to see that $t' \leq t + O(k^3)$. □

3.2 The construction of PCD

In this section, we construct a family of Paillier's trap-door permutations with a common domain.

Definition 8 (family of Paillier's trap-door permutations with a common domain). *The specifications of the family of Paillier's trap-door permutations with a common domain* $\mathsf{PCD} = (K, S, E)$ *are as follows. The key generation algorithm is the same as that for* Paillier. $\mathrm{Dom}_{\mathsf{PCD}}(N, k)$ *and* $\mathrm{Rng}_{\mathsf{PCD}}(N, k, \lambda)$ *are both equal to* $\{x_1 + x_2 \cdot N | (x_1 + x_2 \cdot N) \in [0, 2^{2k}),\ x_1 \in \mathbb{Z}_N,\ (x_2 \bmod N) \in \mathbb{Z}_N^*\}$. *The sampling algorithm returns a random point in* $\mathrm{Dom}_{\mathsf{PCD}}(N, k)$. *The evaluation algorithm* $E(x) = F_{\mathsf{PCD}}(x)$, *and the inversion algorithm* $I(y) = G_{\mathsf{PCD}}(y)$ *are as follows.*

```
Function F_PCD(x)
  u ← F^1_PCD(x); v ← F^2_PCD(u); y ← F^3_PCD(v)
  return y

Function F^1_PCD(x)
  if (x < N^2) u ← F_P(x) else u ← x
  return u

Function F^2_PCD(u)
  if (u < 2^2k − N^2) v ← u + N^2
  elseif (2^2k − N^2 ≤ u < N^2) v ← u
  else v ← u − N^2
  return v

Function F^3_PCD(v)
  if (v < N^2) y ← F_P(v) else y ← v
  return y

Function G_PCD(y)
  v ← G^1_PCD(y); u ← G^2_PCD(v); x ← G^3_PCD(u)
  return x

Function G^1_PCD(y)
  if (y < N^2) v ← G_P(y) else v ← y
  return v

Function G^2_PCD(v)
  if (v < 2^2k − N^2) u ← v + N^2
  elseif (2^2k − N^2 ≤ v < N^2) u ← v
  else u ← v − N^2
  return u

Function G^3_PCD(u)
  if (u < N^2) x ← G_P(u) else x ← u
  return x
```

The choice of N^2 from $(2^{2k-1}, 2^{2k})$ ensures that all elements in $\mathrm{Dom}(F_{\mathsf{PCD}})$ are permuted by F_P at least once. It is clear that F_{PCD} is bijective since F_P is bijective.

3.3 Properties of PCD

In this section, the θ-partial one-wayness of PCD is equivalent to the one-wayness of PCD for $\theta > 0.5$ and the one-wayness of PCD is equivalent to the one-wayness of Paillier.

Theorem 4. *If* PCD *is one-way then* PCD *is* θ*-partial one-way for* $\theta > 0.5$.

Proof. Let A be an algorithm that outputs the $2k - k_0$ most significant bits of the pre-image of its input $y \in \mathrm{Rng}_{\mathsf{PCD}}(N, k)$ with $k > k_0$ (i.e. A is a $((2k - k_0)/2k)$-partial inverting algorithm for PCD with $k > k_0$), with success probability $\epsilon = \mathbf{Adv}^{\theta-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{PCD},A}(k)$ where $\theta = (2k - k_0)/2k > 0.5$, within time bound t. We prove that there exists an algorithm B that outputs a pre-image of y with success probability $\epsilon' = \mathbf{Adv}^{1-\mathrm{pow}-\mathrm{fnc}}_{\mathsf{PCD},C}(k) \geq \epsilon/2$. within time bound $t' \leq t + O(k^3)$. We construct the algorithm B as follows.

Algorithm $B((N,k),w)$

$X \leftarrow A((N,k),w)$
$c \xleftarrow{R} \{0,1\}$
$x_2 \leftarrow ((2^{k_0} \cdot X) \operatorname{div} N) + c$
$w_1 \leftarrow w \bmod N;\ w_2 \leftarrow w \operatorname{div} N$

if $(x_2 \geq N \vee w_2 \geq N)$
  $Y \leftarrow w_1 \cdot N + (w_2 \bmod N)$
  find x_1 s.t. $1 + Nx_1 = \dfrac{Y}{(x_2 \bmod N)^N} \bmod N^2$
else
  $Z \leftarrow w_1 \cdot N + w_2$
  $y_2 \leftarrow x_2^N \bmod N$
  find x_1 s.t.
    $$1 + Nx_1 = \frac{1}{x_2^N} \left[\left(\frac{Z}{y_2^N} - 1\right) + y_2\right] \bmod N^2$$
$x \leftarrow x_1 + x_2 \cdot N$
return x

Analysis

Assume that $w = w_1 + w_2 \cdot N \in \mathrm{Rng}_{\mathsf{PCD}}(N,k)$ and $x = x_1 + x_2 \cdot N = G_{\mathsf{PCD}}(y)$.

B computes x_2 like the inverting algorithm in the proof of Theorem 3.

If $x_2 \geq N$ or $w_2 \geq N$, x is permuted by F_P only once, and then, we have

$$w_1 + (w_2 \bmod N) \cdot N = F_P(x_1 + (x_2 \bmod N) \cdot N).$$

Therefore, we can compute x_1 like the inverting algorithm in the proof of Theorem 3 with replacing x_2 by $x_2 \bmod N$ and w_2 by $w_2 \bmod N$.

If $x_2 < N$ and $w_2 < N$, x is permuted by F_P twice, that is, $w = F_P(F_P(x))$. Assume that $y = y_1 + y_2 \cdot N = F_P(x)$. By the definition of F_P, we have

$$y_1 \cdot N + y_2 = (1 + Nx_1)x_2^N \pmod{N^2}$$

and

$$Z = w_1 \cdot N + w_2 = (1 + Ny_1)y_2^N \pmod{N^2}.$$

Thus,

$$(Ny_1 =)\ (1+Nx_1)x_2^N - y_2 = \frac{Z}{y_2^N} - 1 \pmod{N^2},$$

$$1 + Nx_1 = \frac{1}{x_2^N}\left[\left(\frac{Z}{y_2^N} - 1\right) + y_2\right] \pmod{N^2}.$$

Since $1 + Nx_1 < N^2$,

$$1 + Nx_1 = \frac{1}{x_2^N}\left[\left(\frac{Z}{y_2^N} - 1\right) + y_2\right] \bmod N^2. \quad (1)$$

Furthermore, $y_2 = ((1 + Nx_1)x_2^N \bmod N^2) \bmod N = x_2^N \bmod N$, B can compute the right term of equation 1. Therefore, B can compute x_1.

Hence, if x_2 is correct then $x = x_1 + x_2 \cdot N$ is the pre-image of w, and we have $\epsilon' = \mathbf{Adv}^{1\text{-pow-fnc}}_{\mathsf{PCD},B}(k) \geq \epsilon/2$. It is also clear that $t' \leq t + O(k^3)$. □

It is clear that if PCD is θ-partial one-way then PCD is one-way for $\theta > 0.5$. Thus, the θ-partial one-wayness of PCD is equivalent to the one-wayness of PCD for $\theta > 0.5$.

We can prove the following theorem.

Theorem 5. *If* PCD *is one-way then* Paillier *is one-way.*

It is clear that if Paillier is one-way then PCD is one-way. Thus, the one-wayness of PCD is equivalent to the one-wayness of Paillier.

References

[1] Bellare, M., Boldyreva, A., Desai, A., and Pointcheval, D. Key-Privacy in Public-Key Encryption. In *Advances in Cryptology - ASIACRYPT 2001* (Gold Coast, Australia, December 2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Conputer Science*, Springer-Verlag, pp. 566–582.

[2] Bellare, M., and Rogaway, P. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Advances in Cryptology - EUROCRYPT '94* (Perugia, Italy, May 1994), A. D. Santis, Ed., vol. 950 of *Lecture Notes in Conputer Science*, Springer-Verlag, pp. 92–111.

[3] Fujisaki, E., Okamoto, T., Pointcheval, D., and Stern, J. RSA-OAEP is Secure under the RSA Assumption. In *Advances in Cryptology - CRYPTO 2001* (Santa Barbara, California, USA, August 2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Conputer Science*, Springer-Verlag, pp. 260–274.

[4] Paillier, P. Public-Key Cryptosystems Based on Conposite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99* (Prague, Czech Republic, May 1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Conputer Science*, Springer-Verlag, pp. 223–238.